ECE 571 – Advanced Microprocessor-Based Design Lecture 12

Vince Weaver

https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

30 September 2024

Announcements

- HW#4 was posted. Branch Prediction.

Caches

"Almost all programming can be viewed as an exercise in caching." – **Terje Mathisen**

First Data Cache: IBM System/360 Model 85, 1968

Good survey paper, Ajay Smith, 1982

Computer Architects don't like to admit it, but no amazing breakthroughs in years. Mostly incremental changes.

What is a cache?

- Small piece of fast memory that is close to the CPU.
- “caches” subsets of main memory
- Managed automatically by hardware (can you have a software controlled cache? Scratchpad memory? Why aren’t they used more? Hard to do right.)

Memory Wall

- Processors getting faster (and recently, more cores) and the memory subsystem cannot keep up.
- Modern processors spend a lot of time waiting for memory
- "Memory Wall" term coined by Wulf and McKee, 1995

Exploits Program Locality

- Temporal – if data is accessed, likely to be accessed again soon
- Spatial – if data is accessed, likely to access nearby data

Not guaranteed, but true more often than not

Memory Hierarchy

There's never enough memory, so a hierarchy is created of increasingly slow storage.

- Older: CPU $\rightarrow$ Memory $\rightarrow$ Disk $\rightarrow$ Tape
- Old: CPU $\rightarrow$ L1 Cache $\rightarrow$ Memory $\rightarrow$ Disk
- Now?: CPU $\rightarrow$ L1/L2/L3 Cache $\rightarrow$ Memory $\rightarrow$ SSD Disk $\rightarrow$ Network/Cloud

Cache Types

- Instruction Cache (I$)
  - holds instructions
  - often read only (what about self-modifying code?)
  - can hold extra info (branch prediction hints, instruction decode boundaries)
  - Trace cache variant
- Data Cache (D$) – holds data
- Unified Cache
  - holds both instruction and data

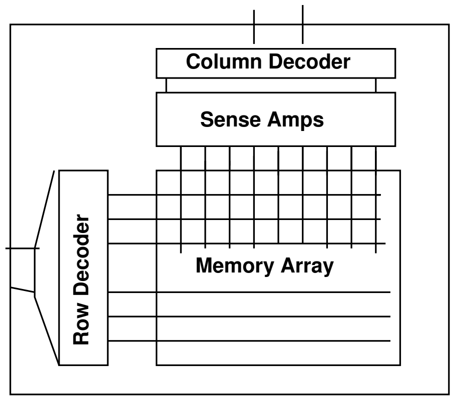
◦ More flexible than separate

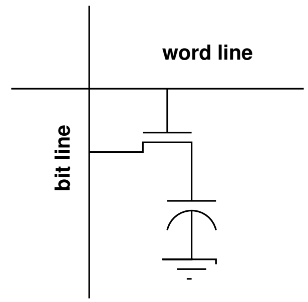
Cache Circuitry

- SRAM – flip-flops, not as dense
- DRAM – fewer transistors, but huge capacitors
  chips fabbed in DRAM process slower than normal CPU logic

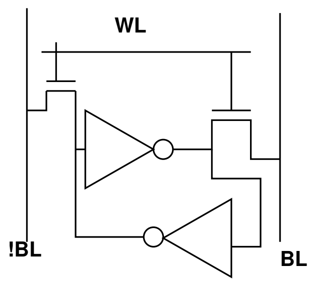
SRAM/DRAM Circuitry

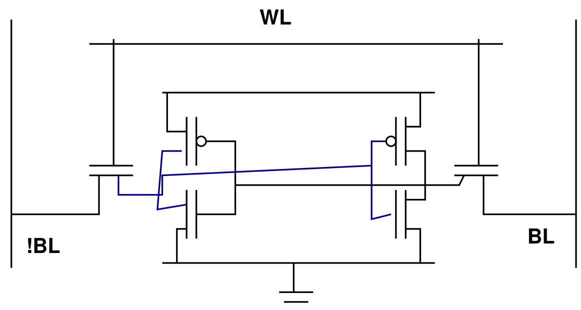
DRAM





SRAM





SRAM/DRAM Benefits

- Upside of DRAM? Smaller, can fit more.
- Upside of SRAM? No need to refresh.
- Which is faster/lower energy? Used to be SRAM but not so clear anymore.
- Why not use DRAM in caches? Process tech doesn't line up well. Process for good capacitors makes slower logic.
- Recent advances (trench capacitors, etc) have changed this a bit. IBM Power machines with large DRAM

caches.

How to see if data in cache

- In addition to data, store address
- For 1k cache should you have 1k addresses?
  Usually you have blocks of bytes, not just one byte
  If (for example) 16-bytes on 32-bit system then only need 28 bits of tag
- A cache like this with just tags and data and data can go anywhere is called fully-associative

Cache Associativity – Fully Associative

- Also known as content-addressable memory (CAM)
- an address can map to any cache line
- Downside: need to compare tag against all entries
  - This either takes lots of comparators (area/power)
  - Or can have 1 comparator but take lots of time

Cache Associativity – Direct Mapped

- Instead can use a few bits from address to separate addresses to "lines"
- Sort of like how we did branch predictors
- Only need one comparator to check tag (so fast / low area)
- Downside: can have aliasing if multiple addresses we need map to the same line

Cache Associativity – Set Associative

- Compromise between direct mapped and fully associative
- Map to a cache line, but there can be multiple "ways"
- For a 4-way cache, you need to check 4 tags (doable)
- Helps avoid aliasing problems
- Can still have wasted space in cache (compared to full) if for some reason some addresses don't map evenly

Cache Structure

	Way 0					Way 1				
line	Tag	Data				Tag	Data			
0	1	00	00	00	00	0				
1	1	00	10	00	00	1	00	00	00	00
2	0					0				
3	0					0				
4	0					0				
5	0					0				
. . .										
b	0					0				
c	0					0				
d	0					0				
e	0					0				
f	0					0				

Cache Terms

- Line – which row of a cache being accessed
- Blocks – size of data chunk stored by a cache
- Tags – used to indicate high bits of address; used to detect cache hits
- Sets (or ways) – parts of an associative cache

What to do on Cache Miss?

- If miss in cache, need to bring in the missing data
- If existing data is there need to evict it
- How can you tell if valid data there? (valid bit)
- In set-associative cache, how do you pick which way to evict?

Replacement Policy

- FIFO
- Least Recently Used (hard to track when way count gets large)
- Round-robin
- Random (Surprisingly effective)
- Pseudo-LRU (not-most-recently-used?)
- Spatial

Load Policy

- Critical Word First – when loading a multiple-byte line, bring in the bytes of interest first

Stores are a Problem

- As with memory order in pipelines, stores are much more of a pain than loads are

Consistency

Need to make sure Memory eventually matches what we have in cache.

- write-back – keeps track of dirty blocks, only writes back at eviction time. poor interaction on multi-processor machines
- write-through – easiest for consistency, potentially more bandwidth needed, values written that are discarded
- write-allocate – Usually in conjunction with write-back Load cacheline from memory before writing.

Inclusiveness

- Inclusive – every item in L1 also in L2
  simple, but wastes cache space (multiple copies)

- Exclusive – item cannot be in multiple levels at a time